

Umfassende Einführung in die mobile Datenkommunikation per WAP am Beispiel der Entwicklung eines WAP-Service

Schwerpunkte:

- WAP Grundlagen
- Programmierung von WAP-Seiten in PHP
- Sessionmanagement in WAP
- Usermanagement
- MySQL Datenbankbindung

Jens Bierkandt
bierkandt@waptune24.de
<http://www.waptune24.de/>

21.06.2001, Version 1.1

Inhaltsverzeichnis

1	Einleitung	7
2	Grundlagen	9
2.1	Was ist das Wireless Application Protocol (WAP)	9
2.2	Wer braucht WAP	9
2.3	Wie funktioniert WAP	10
2.3.1	Anwendungsschicht	11
2.3.2	Sitzungsschicht	12
2.3.3	Übertragungsschicht	13
2.3.4	Sicherungsschicht	13
2.3.5	Transportschicht	13
2.4	Typische WAP-Anwendungen	13
2.5	Übersicht verschiedener WAP-Geräte	15
2.6	Übersicht verschiedener WAP-Emulatoren	15
2.7	Die Zukunft von WAP	16
3	Entwicklung eines WAP-Service	19
3.1	Einführung	19
3.1.1	Allgemeine Vorüberlegungen	20
3.1.2	Technologieauswahl	20
3.1.3	Anpassen des Webservers	21
3.1.4	Die zehn goldenen Regeln	22
3.1.5	Das Sessionmanagement	24
3.2	Lektion 1	25
3.2.1	Die Programmstruktur	25

3.2.2	Die Datenbankstruktur	25
3.2.3	Die Programme	28
3.3	Lektion 2	29
3.3.1	Erstellen des Logos	29
3.3.2	Konfigurationsdatei	30
3.3.3	Die Einstiegsseite	31
3.3.4	Neues Login erstellen	31
3.3.5	Anmelden mit vorhandenem Login	31
3.4	Lektion 3	32
3.4.1	Neues Login überprüfen	32
3.4.2	Darstellung der Linkseite	32
3.4.3	Darstellung der Textinformationen	32
3.5	Lektion 4	33
3.5.1	Hinzufügen von Daten	33
3.5.2	Löschen von Daten	33
4	Mobile Übertragungsstandards	35
4.1	GSM	35
4.2	GPRS	35
4.3	UMTS	36
4.4	LBS (Location based services)	36
4.5	Bluetooth	37
4.6	VoiceXML	37
4.7	i-mode	37
5	Zusammenfassung	39
A	Der Programmcode	43
A.1	index.php4	43
A.2	config.php4	44
A.3	info.php4	46
A.4	login.php4	47
A.5	newlogin.php4	48
A.6	newlogindo.php4	49

A.7	links.php4	51
A.8	text.php4	53
A.9	insert.php4	54
A.10	insertcheck.php4	55
A.11	deldata.php4	57
A.12	maketable.sql	58

Kapitel 1

Einleitung

Als vor einigen Jahren das Internet seinen Siegeszug begann war sicher den wenigsten Menschen bewusst wie umfassend es unser Leben verändern würde. Briefe werden inzwischen in Sekundenschnelle als e-Mails übertragen, beliebige Informationen können aus einer schier unglaublichen Datenmenge in Echtzeit gesucht werden. Arbeiten, die bisher außer Haus erledigt werden mussten, können heute bequem vom Schreibtisch aus getätigt werden.

Viele Menschen möchten diesen Komfort heute nicht mehr missen, egal, wo sie sich gerade aufhalten. Bisher musste zur Nutzung des Internet und zum Datenaustausch immer ein Computer benutzt werden. Erst mit der Einführung von mobilen Geräten wie Handys konnten Personen überall erreichbar sein und von überall mit Personen kommunizieren. Auch konnten schon kleine e-Mails an andere mobile Teilnehmer geschrieben werden, ermöglicht durch den Short Messaging Service, kurz SMS. Heute sind diese kleinen Nachrichten für viele Menschen nicht mehr wegzudenken, speziell nicht für kaufkräftige Kids, die am Handy-Boom nicht ganz unbeteiligt sind.

Doch es hat sich gezeigt, dass der Markt mehr verlangt. Vor allem Berufsgruppen, die immer auf einem aktuellen Stand sein müssen (z.B. Broker oder Manager), verlangen nach mobiler Informationsvielfalt. Dies wurde schnell von den marktführenden Handy-Herstellern erkannt. So hat sich Ende 1997 das WAP-Forum [1] gegründet, initiiert von Motorola, Ericsson, Nokia und Unwired Planet (phone.com). Diese setzten sich zum Ziel, den gemeinsamen Standard "WAP" für

mobile Datenübertragung zu schaffen.

WAP bedeutet den Schritt zum mobilen Internet. Er ist für Handys mit Ihren bisher beschränkten Fähigkeiten vergleichbar wie der Unterschied zwischen einem Telex und einem Fax. Jetzt ist es möglich, mobil, d.h. von fast jedem Standort aus, Daten mit der Welt auszutauschen. Aktienkurse können in Echtzeit abgerufen werden, der Zugfahrplan ist mobil verfügbar. . . Handys sind jetzt nicht mehr nur Telefone sondern multifunktionale Kleincomputer, auf denen Applikationen ausgeführt werden können und die mit anderen Geräten auf Datenebene kommunizieren. Auch haben sich neue Dienste wie das Unified Messaging für mobile Geräte entwickelt. Hier werden FAX, e-Mail, Adressbuch und weitere Softwaremodule zusammengefasst und für WAP-Geräte angeboten.

Das Internet mutiert zum Evernet.

Kapitel 2

Grundlagen

2.1 Was ist das Wireless Application Protocol (WAP)

WAP steht als Abkürzung für das Wireless Application Protocol. Es wurde 1997 vom WAP-Forum spezifiziert und wird ständig von dessen Mitgliedern, dem Hersteller, Netzbetreiber und Webentwickler angehören, überwacht und weiterentwickelt. Die Spezifikation definiert verschiedene Eckwerte, wie mobile Geräte auf Datendienste wie das Internet zugreifen können. So werden z.B. das Übertragungsprotokoll vom Gerät zum Internet oder die Wireless Markup Language (WML), mit dem ähnlich wie bei der Hypertext Markup Language (HTML) die WAP-Seiten programmiert werden, festgelegt. Die aktuelle Version ist zur Zeit 1.2. Allerdings sollte man aus Gründen der Kompatibilität mit älteren Handys lieber auf die Spezifikation 1.1 zurückgreifen.

2.2 Wer braucht WAP

WAP wurde u.a. entwickelt, um die geringen Datenübertragungsraten des GSM-Netzes effizient auszunutzen. Diese betragen 9600 Bit/s, für einen Internetzugang eine unzumutbare Geschwindigkeit. Um mit dieser Bandbreite effektiver umzugehen, komprimiert der WAP-Standard die Daten. Außerdem haben mobile Geräte relativ kleine Displays. Aus diesem Grund wird klar, dass normale HTML-Seiten, welche auf große Bildschirme angepasst sind und eventuell Javascript und

andere Funktionen benutzen, nicht so ohne weiteres auf einem Handy darstellbar sind. Es ergab sich die Notwendigkeit einer neuen Sprache, welche speziell an die Beschränkungen mobiler Geräte angepasst ist. Dies gilt auch für die nicht vorhandene Tastatur.

WAP eignet sich also in erster Linie für mobile Geräte wie Handys und PDAs, die über den GSM Standard oder dessen Nachfolger wie GPRS oder UMTS mit dem Internet kommunizieren. Im Grunde genommen kann aber fast jedes mobile Netz damit genutzt werden.

2.3 Wie funktioniert WAP

WAP ist im Grunde genommen nichts anderes als eine Spezifikation, die die Kommunikation zwischen dem mobilen Gerät wie einem Handy und dem Internetserver, auf welchem die aufgerufenen WML-Seiten liegen, regelt.

Zwischen dem Handy und dem Server liegt ein Gateway. Dieses ist für den nahtlosen Übergang von der mobilen Übertragungsschiene zum Internet zuständig. Diesem Gateway (auch WAP-Proxy genannt) fällt somit die Aufgabe zu, zwischen dem Telefonnetz und dem Internet zu übersetzen. Vom Gateway zum Internetserver werden hier die Protokolle auf HTTP und TCP/IP umgesetzt und die dann erhaltenen Daten vom Gateway zum Handy binär codiert und komprimiert, um die Datenmengen zu verkleinern und Bandbreite zu sparen. Abbildung 2.1 verdeutlicht diesen Zusammenhang.

Das WAP-Gerät stellt durch den Microbrowser WML-Seiten dar. Die Sprache ähnelt sehr stark HTML, benutzt aber den XML-Standard. Außerdem bietet sie ein paar nützliche Funktionen, welche den Umgang beim Browsen mit dem WAP-Handy vereinfachen. Daneben gibt es WMLScript als Skriptsprache, welche mit Javascript vergleichbar ist.

Damit WAP effizient arbeitet, wurde ein spezielles Schichtenmodell eingeführt. Ähnliche Modelle kennt man von anderen Netzwerkprotokollfamilien wie TCP/IP. WAP wird durch fünf Schichten beschrieben. Dies ist in Abbildung 2.2 dargestellt.

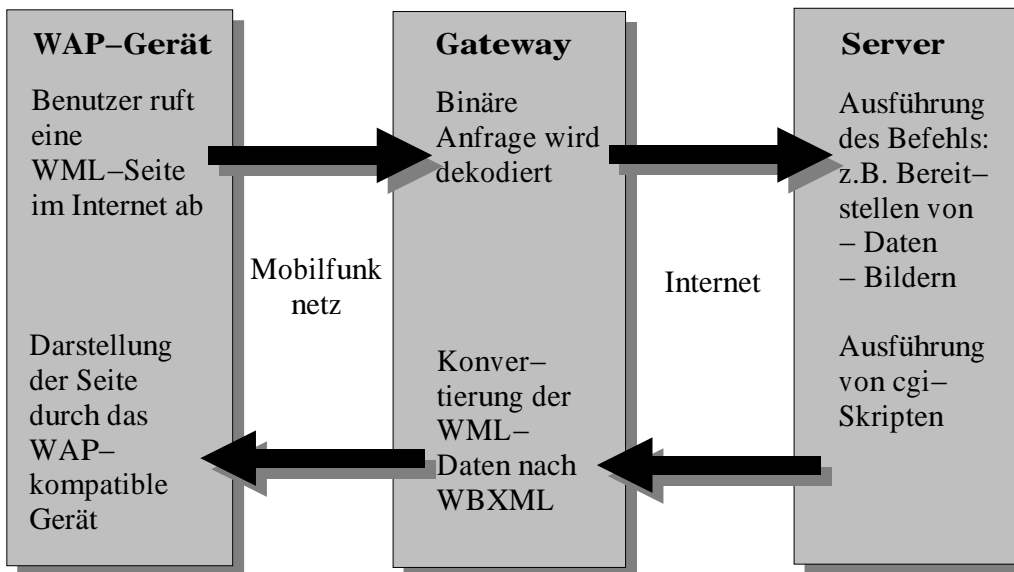


Abbildung 2.1: Der Übertragungsweg einer WAP-Anfrage

1. Anwendungsschicht (Application Layer):
Wireless Application Environment (WAE)
2. Sitzungsschicht (Session Layer):
Wireless Session Protocol (WSP)
3. Übertragungsschicht (Transaction Layer):
Wireless Transaction Protocol (WTP)
4. Sicherheitsschicht (Security Layer):
Wireless Transport Layer Security (WTLS)
5. Transportschicht (Transport Layer):
Wireless Datagram Protocol (WDP)

2.3.1 Anwendungsschicht

Die Anwendungsschicht (Wireless Application Environment, WAE) dient in erster Linie als Ausführungsumgebung von WAP-Anwendungen auf WWW- und



Abbildung 2.2: Das Schichtenmodell von WAP

Mobiltelefonbasis. Dazu gehören u.a.

- ein Microbrowser, der die WAP-Inhalte darstellt
- die Wireless Telephony Application (WTA)
- eine Interfacebeschreibung, um Telephoniedienste zu programmieren
- ein Set von Datenbeschreibungen (Content Formats) für Bilder und Telefonbucheinträge

2.3.2 Sitzungsschicht

In der Sitzungsschicht (Wireless Session Protocol, WSP) werden zwei verschiedene Dienste angeboten. Ein verbindungsorientierter Dienst, der oberhalb des WTP operiert und die Schnittstelle zum verbindungslosen Dienst WDP, der als Datagramm-Service bereitsteht.

Zur Zeit wird das WSP/B Protokoll benutzt. Es dient dazu, Dienste für Browser zur Verfügung zu stellen und HTTP/1.1 Funktionalität über stark verzögernde und langsame Netzwerke, wie sie bei mobiler Kommunikation auftreten, bereitzustellen.

2.3.3 Übertragungsschicht

Die Übertragungsschicht (Wireless Transaction Protocol, WTP) garantiert bei Bedarf die als zuverlässig deklarierten Transaktionen. Dies kann zum Beispiel für Authentifizierungen ausgenutzt werden.

2.3.4 Sicherungsschicht

Die Sicherungsschicht (Wireless Transport Layer Security, WTLS) basiert auf der Transport Layer Security (TLS), besser bekannt aus dem WWW unter dem Namen Secure Socket Layer (SSL). Sie sorgt für die Authentizität, Datenintegrität und Vertraulichkeit der übermittelten Daten. Darüber hinaus bietet sie einen gewissen Schutz gegen Denial-of-Service Attacken.

2.3.5 Transportschicht

Die Transportschicht (Wireless Datagram Protocol, WDP) ist für die Kommunikation zwischen dem Bearer und den darüber liegenden Schichten zuständig. Als Bearer bezeichnet man die Schnittstelle zwischen WAP und physikalischen Netzen wie GSM oder TCP/IP-Netzen, hiermit wird die Kommunikation zwischen dem Gateway und dem WAP-Gerät hergestellt. Es ist wie das von TCP/IP bekannte Protokoll UDP verbindungslos.

2.4 Typische WAP-Anwendungen

Bei der Entwicklung von Anwendungen für mobile Geräte sollte nicht vergessen werden, dass der Benutzer nur an Daten interessiert ist, die er unterwegs braucht. Dies können z.B. aktuelle Informationen sein, welche Abfragen aus einer umfangreichen Datenbank beinhalten. Tabelle 2.1 zeigt einen kleinen Überblick vorhandener Angebote.

Dienstleistung	WAP-Adresse
WAP-Portalseite von Yahoo. Zugriff auf persönlichen Kalender, Adressbuch und e-Mail	wap.yahoo.de
Umfangreiche Suchmaschine für WAP-Seiten	wap.fireball.de
Ermöglicht das Suchen von Auktionen in der Datenbank von e-bay	wap.ebay.de
Bietet die Suche nach einem Hotel in fast jeder größeren Stadt	wap.hoteldirect.de
Der Zug hat Verspätung? Hier können Sie direkt ihre neue Verbindung erfahren	wap.bahn.de
Der Hersteller des Routenplaners Map&Guide bietet hier schnelle Hilfe für unterwegs	wap.cas.de
Der ADAC informiert auf seinen WAP-Seiten über aktuelle Staus und Verkehrsstörungen	wap.adac.de
Hier können Aktiengeschäfte unterwegs online getätigt werden	wap.consors.de
Tagesaktuelle Aktienempfehlungen und Unternehmensnachrichten	www.teleboerse-tipp.de
Das Neueste aus der Welt kompakt zusammenfasst	wap.tagesschau.de
Wettervorhersage, astronomische Daten, Biowetter und Pollenfluginformationen	wap.wetteronline.de

Tabelle 2.1: Typische WAP-Anwendungen

Alcatel One Touch 501	Nokia 7110
Alcatel OTE 301	Nokia 9110i Communicator
Ericsson R320s	Panasonic GD93
Ericsson R380s	Pillips Xenium
Motorola T2288	Siemens C35i
Motorola Talkabout T2288	Siemens M35i
Motorola Timeport 250	Siemens S35i
Motorola Timeport P7389	Siemens SL45
Motorola Timeport V50	Sony CMD Z5
Motorola V2288	Sony CMD J5
Nokia 6210	

Tabelle 2.2: Aktuelle WAP-Geräte

2.5 Übersicht verschiedener WAP-Geräte

Nachdem es kurz nach der Einführung des WAP-Standards sehr wenige WAP-fähige Geräte auf dem Markt gab, hat sich die Lage inzwischen doch deutlich verbessert. Tabelle 2.2 zeigt eine Übersicht über zur Zeit übliche WAP-Geräte.

2.6 Übersicht verschiedener WAP-Emulatoren

Bei der Entwicklung eines WAP-Service müssen die WML-Seiten sehr oft überprüft und getestet werden. Um die Kosten zu senken und den Bedienkomfort zu steigern, wurden Browser entwickelt, welche unabhängig von einem WAP-Gerät WML-Seiten darstellen können. Man spricht hierbei von WAP-Emulatoren. Diese sind online verfügbar, d.h., es wird eine Internetverbindung benötigt und die Ergebnisse können meist in einem normalen HTML-Browser betrachtet werden. Es können aber auch Offline-Browser benutzt werden, die als eigenständige Programme auf dem Computer laufen und meist bessere Ergebnisse liefern. Tabelle 2.3 und Tabelle 2.4 zeigen eine Auswahl brauchbarer WAP-Emulatoren.

Zu beachten ist allerdings, dass ein Browser nie perfekt ein WAP-Gerät emulieren kann. So kann es vorkommen, dass Seiten auf einem realen WAP-Gerät funktionieren, auf einem Online-Browser aber nicht, oder umgekehrt. Deshalb sollte eine fertige Applikation immer direkt auf den Handys und PDAs getestet

Online Browser	Homepage	Beschreibung
iobox	www.iobox.de	Die WML-Befehle sind weitgehend umgesetzt
Wapalizer	www.gelon.net	Relativ ausgereifter Browser mit gutem Handydesign
Wapjag	www.wapjag.de	Ein weiterer gelungener Emulator, mit Ansicht des WML-Quellcodes
Wapsilon	www.wapsilon.com	Emuliert Fullscreen, Nokia 7110 und einen PDA
Yahoo!	cgi.europe.yahoo.com /de/wap/wap.cgi	Rudimentäre Funktionen, nur für einen schnellen Überblick geeignet
WAP-Tiger	www.waptiger.de	Gewöhnungsbedürftig, einige Befehle fehlen

Tabelle 2.3: Online Browser

werden.

2.7 Die Zukunft von WAP

Nichts wird in einschlägigen WAP-Diskussionsforen [2] so heftig diskutiert wie die Zukunft von WAP. Auf der einen Seite gibt es viele Personen, die diesem Protokoll keine Überlebenschance geben. Auf der anderen heftige Verfechter des Standards und überzeugte WAP-Anhänger. Wer letztendlich Recht haben wird, kann heute noch nicht gesagt werden. Eines ist allerdings unumstritten: wäre 1997 nicht der WAP-Standard definiert worden, hätten wir bedeutend länger auf Handys mit Internetanschluss warten müssen.

WAP hat den großen Vorteil, dass es Daten automatisch komprimiert. Es besteht kein Grund, weswegen ein HTML-Befehl wie `<a href= ...` als ASCII übertragen werden soll. Dieser Befehl taucht immer wieder auf und so liegt es nahe, ihn binärkodiert zu übertragen. Somit wird deutlich Bandbreite gespart. Desweiteren basiert WML auf XML und ist ausbaufähig. Für neue Funktionen wird einfach ein neuer Befehl hinzugefügt.

Es ist zwar mit ein paar Hilfsmitteln möglich, HTML-Seiten nach WML zu konvertieren, doch ergibt dies wenig Sinn. Ein WAP-Handy hat nur ca. 15 Tasten

Programm	Betriebs- system	URL	Beschreibung
AU Systems WAP Browser	Palm OS 3.3	www.wapguide.com	WAP Browser für den Palm
Ericsson R380 Emulator	Windows	www.symbian.com	Ericsson R380 Emula- tor
EzWAP	Windows, Pocket PC	www.ezos.com	Gutaussehender Multiplattform- Browser
Nokia WAP Tool- kit	Windows, Linux	www.nokia.com	Gutes Entwicklun- gs- tool
Opera	Windows, Linux, Be- OS, EPOC, MacOS	operasoftware.com	HTML Browser mit eingebautem WML- Emulator
Phone.com UP.SDK	Windows	www.phone.com	Bekannter WML- Browser, in vielen Handys implementiert
WAPman	Windows	www.wap.com.sg	Guter Browser mit ei- genen Skins
WinWAP	Windows	winwap.org	Einer der ersten Brow- ser
YOURWAP.com	Windows	www.yourwap.com	Umfangreiche Simula- tion verschiedener Han- dys

Tabelle 2.4: Offline Browser

und ein kleines Monochromdisplay. Deshalb ist WML zur Zeit die beste Lösung, weil der Standard ideal an mobile Geräte angepasst ist.

Die Handys der Zukunft werden kaum mehr etwas mit den bisherigen Geräten gemeinsam haben. Sie werden zu komplexen Multifunktionsdevices ausgebaut sein, ein wenig Computer, ein wenig Telefon, ein wenig Tastatur (eventuell als Touch-Screen), alles in einem handlichen Gerät vereint. Durch die jetzt schon geplanten Datenübertragungsraten von UMTS und der nachfolgenden Systeme wird es auch möglich sein, diese Geräte multimedialfähig auszubauen und z.B. unterwegs fernzusehen.

Ob WAP dann noch eine Rolle spielt, hängt von vielen Faktoren ab. Eines steht aber fest: der Inhalt für mobile Geräte muss immer an diese angepasst werden. Und hier eignet sich WAP sehr gut.

Kapitel 3

Entwicklung eines WAP-Service

3.1 Einführung

Informationen über die Programmierung von statischen WAP-Seiten gibt es ausreichend [3], [4], [5], [6]. Doch diese dienen meist nur der Grundlagenvermittlung. Sollen Anwendungen geschaffen werden, die nicht nur statischen Quellcode beinhalten und auch professionellen Ansprüchen genügen, bedarf es dynamischen Programmcodes mit Anbindung an eine Datenbank.

Anhand eines Beispiels sollen diese Techniken Schritt für Schritt erklärt werden. Es wird eine Anwendung entwickelt, bei dem der mobile Surfer nach einer Anmeldung Passwörter oder ähnliches in eine Datenbank ablegen kann. Folgende Schritte sind dabei interessant:

1. Usermanagement
2. Datenbankanbindung
3. Sessionmanagement
4. Dynamische Generierung von WML-Seiten

Unter <http://www.waptune24.de/pass/> kann das Programm “*Passwortdatenbank*” per WAP-Gerät oder mit einem Emulator (siehe Seite 15) ausgiebig getestet werden. Beachtet werden sollte, dass das Beispiel nur zu Lehrzwecken dient. Es ist

sehr offen angelegt und kann schnell an beliebige Anwendungen angepasst werden.

3.1.1 Allgemeine Vorüberlegungen

Mit einem WAP-Service sollen vor allem Kunden angesprochen werden, die gerade mobil unterwegs sind und Informationen benötigen. Meistens sind dies aktuelle Daten (z.B. Stauinformationen) oder aber statische aus einer sehr großen Datenbank (z.B. Routenplaner). Inhalte, die bequemer von zu Hause aus abgerufen werden können, sollte man in sein WAP-Angebot nicht aufnehmen.

Weiterhin dürfen nicht die Besonderheiten von WAP-Geräten vergessen werden. So sollte immer an die Einschränkungen wie das kleine monochrome Display, die umständliche Dateneingabe, die längeren Wartezeiten beim Seitenabruf und den noch ziemlich hohen WAP-Minutenpreis gedacht werden. Es sollte bei der Informationsauswahl darauf geachtet werden, dass nur das Nötigste übertragen wird.

Es hat sich eingebürgert, dass beim Aufruf eines WAP-Angebotes als erstes immer dessen Zweck, eventuell mit einem kleinen Logo, erscheint. Danach werden die Benutzer (eventuell automatisch über einen Timer) zu einer Linkliste oder zu Eingabefeldern geleitet, über die schnell die gesuchten Informationen abgerufen werden können oder sich eingeloggt werden kann.

3.1.2 Technologieauswahl

Im Prinzip ist es möglich, WML-Seiten ähnlich wie HTML-Seiten statisch zu programmieren. Dabei werden jedoch eine Menge an Möglichkeiten vergeben, die für aktuelle und umfangreiche Seiten unumgänglich sind. Besser sind hier Sprachen, die beim Abruf einer Seite auf dem Webserver erst Quellcode ausführen. Eine typische Anwendung fragt z.B. eine Datenbank ab und füllt mit dem Ergebnis automatisch eine WML-Seite.

Die Beispielanwendung ist für einen LAMP-Server entwickelt worden. LAMP steht für **L**inux als Betriebssystem, **A**pache für den Webserver, **M**ySQL für die freie MySQL-Datenbank und **P**HP für die Skriptsprache. Die Installation eines

Beschreibung	MIME-Typ	Dateiendung
WML-Datei	text/vnd.wap.wml	.wml
WML-Script	text/vnd.wap.wmlscript	.wmls
WML kompiliert	application/vnd.wap.wmlc	.wmlc
WML-Script kompiliert	application/vnd.wap.wmlscript	.wmlsc
WBMP-Grafik	image/vnd.wap.wbmp	.wbmp

Tabelle 3.1: Die MIME-Typen für WAP

LAMP-Systems kann unter [7] nachgelesen werden und würde den Rahmen dieser Ausführung übersteigen. Auch für PHP gibt es genügend Informationen [8], [9]. Uns interessiert hier mehr die Migration eines LAMP-Systems mit WAP.

Das Beispiel wurde auf folgendem System entwickelt:

- Linux 2.2.18
- Apache 1.3.14
- MySQL 3.22.32
- PHP 4.0.4pl1

Natürlich können auch beliebige andere Applikationen verwendet werden, wie z.B. der Microsoft Information Server oder Perl als Skriptsprache. Doch bietet PHP einen sehr einfachen Zugriff auf eine MySQL-Datenbank und viele Provider benutzen Linux und den Apache als Webserver. Hilfe hierfür und bei anderen WAP spezifischen Fragen bietet eine ausgezeichnete F.A.Q. unter [10].

3.1.3 Anpassen des Webservers

Nachdem das LAMP-System läuft, muss man den Webserver noch für Anfragen von WAP-Geräten anpassen. Hierfür werden verschiedene MIME-Typen (siehe Tabelle 3.1) in die Konfigurationsdatei des Webservers eingetragen.

Die meisten Provider haben dies schon erledigt. Ist das nicht der Fall gibt es die Möglichkeit, die Daten auch in die *.htaccess* Datei im WAP-Verzeichnis abzulegen:

```
# MIME-Types for WAP in httpd oder .htaccess
AddType text/vnd.wap.wml .wml
AddType text/vnd.wap.wmlscript .wmlscript
AddType application/vnd.wap.wmlc .wmlc
AddType application/vnd.wap.wmlscript .wmlsc
AddType image/vnd.wap.wbmp .wbmp
```

Für WAP-Dateien sollte grundsätzlich ein eigenes Verzeichnis gewählt werden. Besteht die Möglichkeit, eine eigene Subdomain einzurichten, ist dies vorzuziehen. Heißt das Webangebot z.B. *http://www.<Adresse>.de/*, so sollte das WAP-Angebot unter *http://wap.<Adresse>.de/* zu finden sein. Damit der Webserver beim Aufruf dieser Adresse auch die Startseite zurückgibt, muss die Subdomain mit dem WAP-Verzeichnis verlinkt sein. Außerdem sollte folgende Zeile in die dortige *.htaccess*-Datei eingetragen werden:

```
DirectoryIndex index.wml
```

Sie bewirkt, dass die Datei *index.wml* als Default-Seite aufgerufen wird. Somit kann nur mit der Angabe des Pfades in der URL die Startseite aufgerufen werden.

3.1.4 Die zehn goldenen Regeln

Um die häufigsten Fehler bei der Entwicklung eines WAP-Services zu vermeiden, wurden folgende einfache Regeln zusammengetragen.

1. Alle Tags werden im WML-Code klein geschrieben.
2. Alle Tags müssen geöffnet und geschlossen werden (z.B. `<p> Text </p>`). Ist ein Tag alleinstehend, so muss ein Slash als Abschluss hinzugefügt werden (z.B. `
`).
3. Die WML-Browser reagieren im Allgemeinen sensibel auf Fehler. Deshalb sollte sehr sorgfältig programmiert werden.
4. Jede Seite sollte vor einer Veröffentlichung auf so vielen WAP-Geräten wie möglich getestet werden. Jedes WAP-Gerät hat seine Besonderheiten und

interpretiert den WML-Standard meist anders. Einige Geräte unterstützen manche Befehle auch gar nicht (z.B. Tabellen).

5. WML kennt keine Umlaute und Sonderzeichen. Diese müssen in Unicode im WML-Quelltext stehen. In unserem Beispiel wurde eine Funktion in PHP geschrieben, die dies automatisch erledigt. Im Abschnitt A.2 auf Seite 44 steht der Quellcode. Eine Erklärung steht auf Seite 30.
6. Jedes Handy besitzt einen Cache um Seiten zwischenspeichern. Manche Handys halten so Seiten bis zu 30 Tage im Speicher und überprüfen bei einem erneuten Aufruf nicht auf aktualisierte Daten auf dem Webserver. Für aktuelle Informationen ist dies nicht tragbar. Deshalb sollte immer der Cache auf die Erfordernisse der Anwendung abgestimmt werden. Zu beachten ist auch, dass jedes Handy unterschiedliche Befehle für das Kontrollieren des Cache hat. Auch hier wurde eine Funktion für unser Beispiel entwickelt, welche bei den meisten WAP-Geräten funktionieren sollte. Siehe Abschnitt A.2 auf Seite 44. Vergleich auch Seite 30.
7. Um das Angebot für möglichst viele WAP-Geräte zugänglich zu machen, sollte es so programmiert werden, dass auch noch ältere Geräte mit weniger Funktionalitäten den Programmcode verstehen. Ein Beispiel ist hier das Nokia 7110, welches eines der ersten WAP-Handys war und immer noch sehr verbreitet ist. Es hat sich eingebürgert, dessen Daten als den kleinsten gemeinsamen Nenner bei der Programmierung von WAP-Seiten nicht zu überschreiten. Tabelle 3.2 zeigt die Beispieldaten des Nokia 7110. Eine andere Möglichkeit besteht darin, für jedes einzelne Handy speziell zugeschnittene Seiten zu programmieren. Doch ist dies sehr aufwendig und lohnt sich meistens nicht.
8. Wenn in einer URL nur ein Verzeichnis angegeben wird, so muss sie mit einem Schrägstrich beendet werden. Zum Beispiel sollte anstatt `http://wap.<adresse>.de` immer `http://wap.<adresse>.de/` geschrieben werden.
9. Bei der Entwicklung eines WAP-Services sollten die Beschränkungen eines WAP-Gerätes nie aus den Augen verloren werden. Dies betrifft das

Ausstattung	Daten
Maximale Größe eines Decks	1397 Bytes
Maximale Größe eines Bildes	1397 Bytes
Maximale Auflösung eines Bildes	95 x 45
Tabellen / Textausrichtung	nein
Zeilen im Display	4 + Titel
Zeichen/Zeile	19

Tabelle 3.2: Beispieldaten des Nokia 7110

kleine Display, die Übertragungsgeschwindigkeit und die kleine Tastatur. Übermitteln Sie per WAP nur das inhaltlich Notwendigste und erwarten Sie keine großen Eingaben von den Benutzern.

- Erwarten Sie nie zu viel von einem WAP-Angebot. Der Service wird zwar bei entsprechender Werbung und Anmeldung bei einschlägigen WAP-Suchmaschinen meist angenommen, die Zugriffszahlen auf WML-Seiten sind aber erfahrungsgemäß deutlich geringer als auf Web-Seiten. Allerdings ist der Wert der Informationen für Personen, die unterwegs sind, entsprechend größer.

3.1.5 Das Sessionmanagement

Seit der Version 4.0 unterstützt PHP das Sessionmanagement. Damit ist es möglich, unterschiedliche Anfragen desselben Clients miteinander in Beziehung zu setzen.

HTTP ist ein verbindungsloses Protokoll. Server und Client beenden die Verbindung nach jedem Seitenaufruf. Es besteht damit nach einer erneuten Abfrage desselben Clients keine Beziehung zur vorherigen. Um aber Werte von z.B. früheren Eingaben weiter verwenden zu können, wird das Sessionmanagement benötigt. Es speichert die Variablen, die z.B. der User eingegeben hat, auf dem Server ab. Zusätzlich wird beim Client entweder ein Cookie gesetzt oder ein String an jeden Link auf der Seite hinzugefügt. Somit kann man die Verbindungen miteinander in Beziehung setzen.

In unserem Beispiel brauchen wir das Management, um einem User, der sich

eingeloggt hat, zu nachfolgenden Seiten Zutritt zu verschaffen. Damit wird vermieden, dass er sich bei jedem Seitenaufruf neu identifizieren muss. Aus diesem Grund werden die Variablen “*user*” und “*pwd*” als Sessionvariablen gesetzt.

Bei WAP wird meist ein String an die Links angehängt, da die meisten Handys/Proxys keine Cookies akzeptieren. Es wird aber zunehmend mehr WAP-Geräte geben, die auch Cookies unterstützen. Das Sessionmanagement prüft erst, ob es Cookies auf dem Device ablegen kann. Ist dies nicht der Fall, wird an die Links ein String gehängt.

3.2 Lektion 1

Bevor das eigentliche Programmieren beginnt ist es sinnvoll, sich ausführlich Gedanken über die Programm- und Datenbankstruktur zu machen. Welche Seiten benötigen welche Daten, wohin führen die Links und welche Fehler müssen sie abfangen. Damit entfällt später einige unnötige Arbeit bei der Fehlersuche. Lektion 1 gibt einen Überblick über das Beispielprogramm “*Passwortdatenbank*”.

3.2.1 Die Programmstruktur

Das Anmelden eines neuen Users wurde der Übersicht wegen recht einfach gehalten. Ein neues Login wird ohne personenbezogene Daten ausgestellt, das Programm prüft nur, ob Username und Passwort eingegeben wurden und der Username nicht schon vergeben ist. Die Loginstruktur zeigt Abbildung 3.1.

Die Programmstruktur nach dem Login zeigt Abbildung 3.2.

3.2.2 Die Datenbankstruktur

Das Designen einer Datenbank ist nicht ganz einfach. Unser Beispiel zeigt eine Anwendung mit kleinen Tabellen und wenig Datendurchsatz auf. Sollen größere Projekte realisiert werden, muss sich ausführlich mit einer Datenbankgestaltung befasst werden [11].

Jeder User muss sich vor der Nutzung des Dienstes anmelden. Dadurch bekommt man den Usernamen und ein Passwort, welche zusammen mit einer Uni-

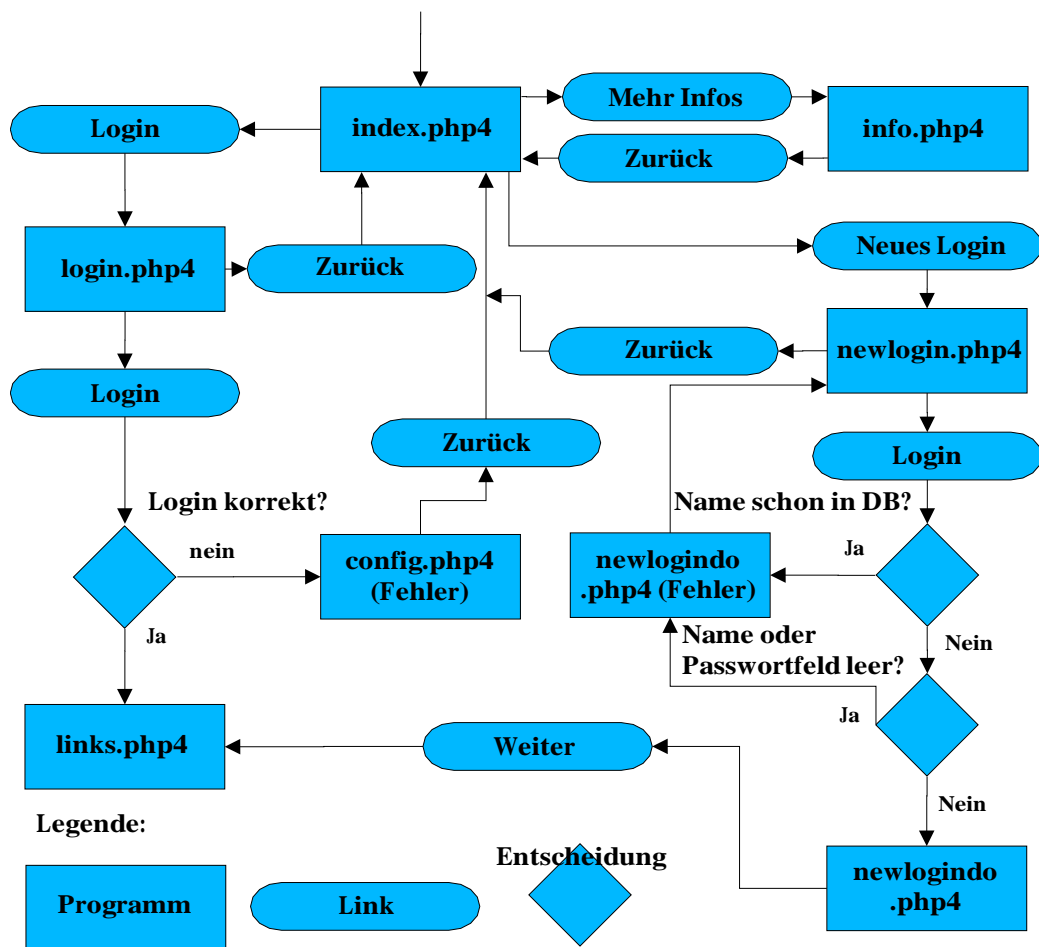


Abbildung 3.1: Programmstruktur Login

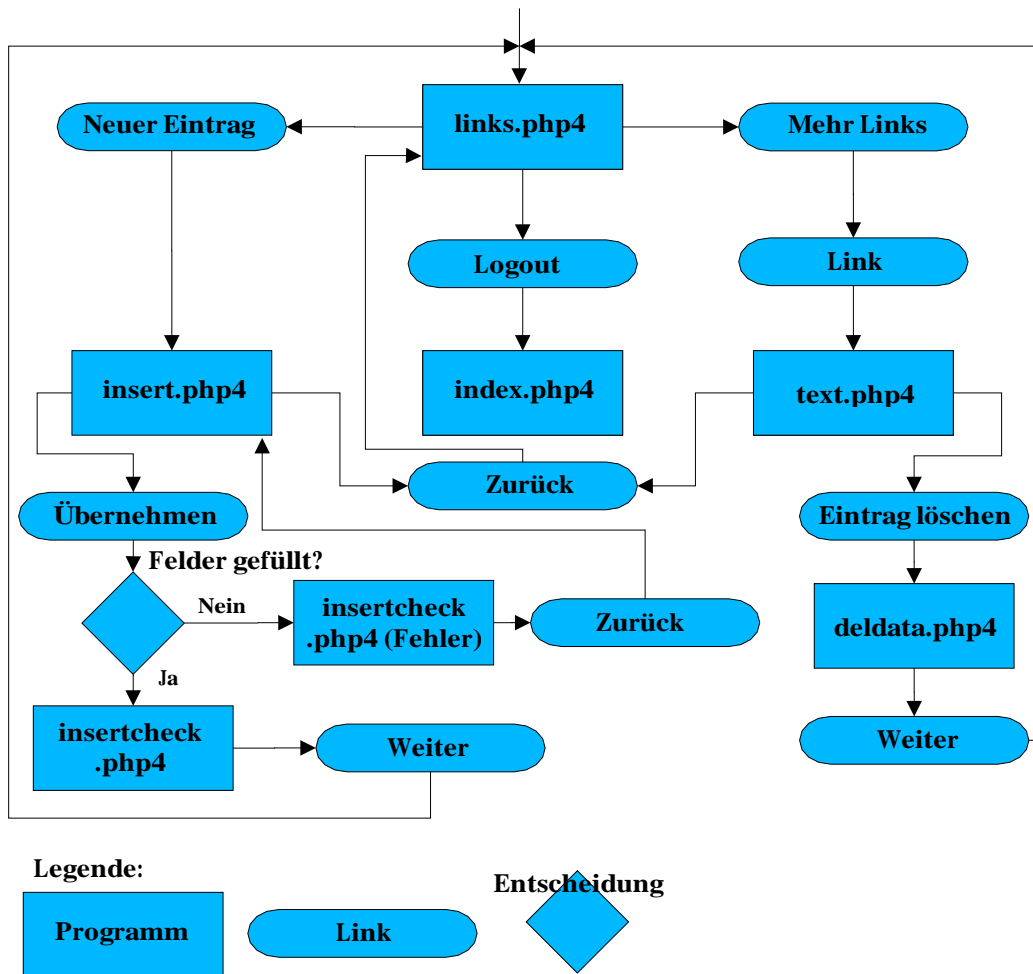


Abbildung 3.2: Programmstruktur nach dem Login

uuid	user	pwd	lastaccess
1	Hans	sdhj&=)8	2001-06-24 15:23:14
2	Peter	HJ@hd;	2001-06-22 11:54:21

Tabelle 3.3: Beispieldaten der Tabelle *passuser*

id	uuid	link	data
1	1	ec-karte	1234
2	1	Passwort Mailbox	467@dgg/
3	2	Telefon Katrin	01234/56789
4	1	Mein Gehalt	geheim
5	2	Passwort Mailbox	df54uje

Tabelle 3.4: Beispieldaten der Tabelle *passdata*

versal User ID (UUID) in die Datenbank geschrieben werden. Die Tabelle kann bei Bedarf erweitert werden, z.B. könnten noch Straße, Ort usw. abgefragt werden. Die UUID dient dazu, in der Tabelle *passdata* die Daten dem User zuzuordnen zu können. Tabelle 3.3 zeigt den Aufbau der Tabelle *passuser*. Es gibt noch ein Datenfeld *lastaccess*, dies wird nur zu Kontrollzwecken für Zugriffe verwendet und nicht an den User ausgegeben.

In der Tabelle *passdata* werden die Daten der User abgelegt. Zwischen *passuser* und *passdata* besteht eine 1:n Abhängigkeit. D.h., ein User kann beliebig viele Daten abspeichern, aber eine Datenreihe ist immer eindeutig einem User zuzuordnen. Tabelle 3.4 zeigt den Aufbau mit Beispieldaten.

Das Skript zur Erstellung der Tabellen steht im Anhang auf Seite 58.

3.2.3 Die Programme

Für jede Funktionalität gibt es ein eigenes Programm. Es überprüft die korrekten Logineingaben, meist aus der Session-ID, und weitere Daten, welche der User eingegeben hat. Danach gibt es eine entsprechende WML-Seite aus.

Das Skript *maketable.sql* ist eine Ausnahme. Es fügt die Tabellen *passuser* und *passdata* automatisch in eine MySQL-Datenbank ein.

Tabelle 3.5 zeigt eine Liste mit den Programmen und für was sie zuständig sind.

Programm	Beschreibung
index.php4	Startseite
info.php4	Zeigt Informationen über den Service
login.php4	Gibt die Loginmaske aus
newlogin.php4	Gibt die Maske zur Eingabe eines neuen Logins aus
newlogindo.php4	Überprüft die Daten des neuen Logins
links.php4	Zeigt eine Liste mit Links zu den Daten an
insert.php4	Maske zur Eingabe von neuen Daten
insertcheck.php4	Überprüft die Eingaben von <i>insert.php</i> und fügt sie in die Datenbank ein
text.php4	Zeigt die Daten eines Links an
deldata.php4	Löscht Daten aus dem Datenbestand
config.php4	Beinhaltet Konstanten und häufig gebrauchte Funktionen
maketable.sql	Fügt automatisch die benötigten Tabellen in die MySQL-Datenbank ein

Tabelle 3.5: Dateien zum Beispiel Passwortdatenbank

3.3 Lektion 2

Nachdem man sich über die Strukturen seines Projektes klar geworden ist, kann schließlich mit der Programmierung begonnen werden. Lektion 2 beschäftigt sich zuerst mit der Programmierung von statischen WML-Seiten.

3.3.1 Erstellen des Logos

Bilder werden beim WAP-Standard im WBMP-Format abgespeichert. Sie sind monochrom und nicht komprimiert. Zur Erstellung eines WBMP-Bildes verwendet man am besten das Programm *pic_2_wbmp* [12], welches sehr viele Grafikformate in das WAP-Format umwandeln kann. Wer seine Bilder online konvertieren lassen möchte, findet in *Terraflops Wireless Bitmap Converter* [13] eine geeignete Alternative.

Gelegentlich müssen auch Bilder dynamisch von Daten aus einer Datenbank erstellt werden (z.B. aktuelle Börsencharts). Hier bietet sich unter PHP die GD-Bibliothek an, welche auch WBMP-Bilder ausgibt. Mit dieser Funktionensammlung kann man grundlegende Zeichenoperationen wie Balken, Kreise, Schrift etc. ausführen.



Abbildung 3.3: Logo des Beispielprogrammes “*Passwortdatenbank*”

Bild 3.3 zeigt das Logo unseres Beispielprogrammes.

3.3.2 Konfigurationsdatei

Es ist bei PHP üblich, Konstanten und häufig gebrauchte Funktionen in eine eigene Datei zu schreiben. In unserem Beispiel heißt sie *config.php4* und ist im Anhang auf Seite 44 zu finden. Neben den Konstanten für die Datenbankverbindung finden sich dort auch drei wichtige Funktionen:

- *headerNoCache()*
- *checkAuth(\$user, \$pwd, \$db)*
- *string2unicode(\$text)*

Die Funktion *headerNoCache* wird zu Beginn jeder WML-Seite aufgerufen. Zuerst wird mitgeteilt, dass es sich bei der darzustellenden Seite um eine WML-Datei handelt. Dann wird das WAP-Handy angewiesen, die Seite nicht im Cache abzuspeichern. Am Schluss wird noch der Anfang der eigentlichen WML-Seite ausgegeben, die Document Type Definition (DTD). Diese ist im WML-Quellcode sichtbar und darf bei keiner WML-Seite fehlen.

Die Funktion *checkAuth* wird bei fast jedem Seitenaufruf aufgerufen. Sie überprüft, ob der User sich eingeloggt hat und ob die Daten korrekt sind. Andernfalls bricht die Funktion mit einer Fehlermeldung ab. Anhand der Fehlerausgabe kann man schon sehr deutlich den grundlegenden Aufbau einer WML-Seite erkennen.

Die Funktion *string2unicode* ist eine der wichtigsten Funktionen. Sie konvertiert gewisse Sonderzeichen und Umlaute in Unicode. Bei diesem Code hat jedes Zeichen eine eigene Zahl und kann so korrekt auf den Browsern, unabhängig vom

gewählten Zeichensatz, dargestellt werden. Es gibt zwar auch wie bei HTML z.B. für das “>”-Zeichen den Ausdruck “>”. Solche Umschreibungen sollten jedoch vermieden werden, da einige WML-Browser damit nicht korrekt umgehen können. Die Funktion testet also die einzelnen Zeichen und ersetzt sie bei Bedarf durch den Unicode.

3.3.3 Die Einstiegsseite

Die Einstiegsseite *index.php4* beinhaltet nur statische Elemente. Das Logo wird angezeigt und man kann sich neben dem Login oder einer Neuanmeldung auch noch Informationen über das Projekt anschauen.

3.3.4 Neues Login erstellen

In unserem Fall ist es den Usern möglich, sich nur mit Angabe eines Usernamens und Passwortes bei unserem Service anzumelden. Normalerweise würde man erst noch weitere (personenbezogene) Angaben machen müssen, aber der Übersichtlichkeit halber wurde dies hier weggelassen. Sobald einmal der Ansatz klar ist, lässt sich das Projekt beliebig erweitern.

Das Programm *newlogin.php4* stellt die benötigte Maske für die Anmeldung bereit.

3.3.5 Anmelden mit vorhandenem Login

Die Logineingabe mit einem bereits vorhandenen Login ist trivial. Das Programm *login.php4* übernimmt diese Aufgabe. Es fragt den Usernamen und das Passwort ab und überträgt es mit der POST-Methode an den Server. Die POST-Methode hängt im Gegensatz zur GET-Methode die Variablen nicht an die URL an sondern stellt sie in den HTTP-Header. Damit werden sie nicht durch den Webserver mitgeloggt, der bei jedem Aufruf einer Webseite die URL mitprotokolliert.

3.4 Lektion 3

Nachdem der User entweder ein schon auf ihn registriertes Passwort eingegeben oder sich neu angemeldet hat, müssen die Daten überprüft werden. Sie werden per POST-Methode an den Server übertragen und dort von unseren Programmen ausgewertet. Diese geben anschließend entsprechende WML-Seiten zurück. Lektion 3 beschäftigt sich mit der dynamischen Ausgabe von WML-Seiten.

3.4.1 Neues Login überprüfen

Nachdem der User sich für ein neues Login angemeldet hat, prüft das Programm *newlogindo.php4*, ob die Daten vollständig sind. Ist dies nicht der Fall, gibt es eine Fehlermeldung aus und bricht ab. Ansonsten schaut es in der Datenbank nach, ob der Username schon vergeben ist. Dazu wird erst eine Verbindung mit der Datenbank hergestellt. Sobald dies erfolgreich war, fragt er nach der UUID des eingegebenen Usernamens. Ist keine vorhanden, gibt die MySQL-Funktion nichts zurück und somit ist der neue Username noch nicht vergeben. Nun wird der neue Name mit Passwort und einer UUID eingetragen. Anschließend gibt das Programm eine Seite aus, die dem User bestätigt, dass er erfolgreich in die Datenbank eingetragen wurde.

3.4.2 Darstellung der Linkseite

Das Programm *links.php4* überprüft wie alle anderen dynamischen Seiten des Beispiels zuerst, ob User und Passwort stimmen. Ist beides korrekt, sieht der Nutzer eine Liste mit Links zu seinen Daten und die Möglichkeit, neue einzugeben. Sind die Daten falsch, wird eine Fehlermeldung ausgegeben und der User kann seine Eingaben auf der Loginseite korrigieren.

3.4.3 Darstellung der Textinformationen

Nachdem der User auf der Seite *links.php4* einen Link ausgewählt hat, kommt er auf die Seite *text.php4*. Dieses Programm gibt zu dem zugehörigen Link die Daten aus. Außerdem erhält der User die Möglichkeit, den Eintrag zu löschen.

3.5 Lektion 4

Lektion 4 beinhaltet Datenbankaufgaben wie das Anlegen und Löschen von Einträgen. Dabei ist es wichtig, sorgfältig zu programmieren, damit der User nicht durch Programmfehler seine mühsam eingegebenen Daten verliert.

3.5.1 Hinzufügen von Daten

Wählt der User auf der Linkseite die Option “Neuer Eintrag”, so kann er auf der Seite *insert.php4* weitere Datenfelder anlegen. Der Linkname wurde auf 15 Zeichen beschränkt, damit er noch auf einer Zeile darstellbar ist. Das Datenfeld kann 500 Zeichen groß sein. Damit ist gewährleistet, dass die Ausgabe des Skripts *text.php4* nicht die maximale Größe von 1397 Bytes überschreitet. Dies würde zu einem Fehler führen. Die Datei *insertcheck.php4* überprüft dann die Eingaben und gibt entweder eine Fehlermeldung zurück oder speichert sie ab.

3.5.2 Löschen von Daten

Das Löschen der Daten wird vom Skript *deldata.php4* ausgeführt. Es überprüft, ob der User eingeloggt ist und löscht entsprechend der Übergabe der Variable *link* von *text.php4* den Datenbankeintrag.

Kapitel 4

Mobile Übertragungsstandards

4.1 GSM

Der “Global Standard for Mobile Communication” (GSM) wurde 1990 ins Leben gerufen. Er spezifiziert zellulare Mobilfunksysteme, die mit digitaler Vermittlungs- und Übertragungstechnik für Sprache und Daten bis 9600 Bit/s je Kanal arbeiten. Weiterhin können Kurznachrichten (SMS) übertragen und umfangreiche zusätzliche Leistungsmerkmale wie bei ISDN (z.B. Anklopfen, Makeln und Dreierkonferenz) benutzt werden. In Deutschland wird hauptsächlich über dieses Netz mobil telefoniert.

Inzwischen haben sich mehr als 130 Länder für diesen Standard entschlossen und betreiben insgesamt über 300 verschiedene GSM-Netze.

4.2 GPRS

Der “General Packet Radio Service (GPRS)” wird zur Zeit in Deutschland eingeführt und basiert auf dem GSM-Standard. Die Daten werden hierbei in kleine Pakete zerlegt und getrennt übertragen. Somit erhöht sich die Bandbreite auf bis zu 53,6 kBit/s (im Vergleich: ISDN hat 64 kBit/s). Weiterhin ist der Handybenutzer ständig online, da im Unterschied zu GSM nur die tatsächlich übertragene Datenmenge berechnet wird. E-Mails können (wie SMS-Nachrichten) sofort zugestellt werden.

Für die Nutzung von GPRS braucht man Handys, die diesen Standard unterstützen, normale GSM-Handys können diese Technik nicht nutzen.

GPRS stellt weiterhin nicht die Übergangslösung von GSM zu UMTS dar. Da UMTS voraussichtlich nicht flächendeckend installiert werden wird, sondern nur in wirtschaftlich attraktiven Regionen, wird man teilweise wieder auf GSM/GPRS zurückgreifen müssen.

4.3 UMTS

Spätestens seit der Versteigerung der “Universal Mobile Telecommunication System (UMTS)“-Lizenzen kennen viele diesen Begriff. Es ist der Nachfolger von GSM und wird voraussichtlich ab 2002 angeboten. UMTS-Verbindungen schaffen im Idealfall die 200-fache Geschwindigkeit von GSM mit bis zu 2MBit/s. Idealfall bedeutet hier, dass sich nicht zu viele Nutzer gleichzeitig in einer Zelle aufhalten dürfen. Außerdem sinkt die Übertragungsrate mit zunehmender Bewegung des mobilen Gerätes. Ähnlich wie bei GPRS wird hier auf den paketorientierten Dienst gesetzt. Der Nutzer ist ständig online und bezahlt nur für die tatsächlich übertragenen Daten.

Interessant ist UMTS wegen der enormen Geschwindigkeitssteigerung. So lassen sich multimediale Elemente wie Filme in Echtzeit übertragen, die langen Wartezeiten der GSM-Technik gehören dann der Vergangenheit an. Das Handy wird mehr und mehr zum multimediafähigen Allround-Talent.

4.4 LBS (Location based services)

Jedes Handy kann anhand der Position der benutzten Funkzelle bis auf einige hundert Meter genau geortet werden. Damit können Dienste mit lokalem Charakter angeboten werden. Stadtspezifische Informationen wie z.B. ein Hotelführer oder das Kinoprogramm können so ohne große Eingaben des Handybenutzers eingeholt werden. Allerdings bestehen datenschutzrechtliche Bedenken, da der Standpunkt des Besitzers ziemlich einfach auszumachen ist. So wird in Pilotprojekten meist explizit angefragt, ob der Benutzer der Standortübertragung zustimmt oder

nicht.

LBS ist zur Zeit noch im Aufbau und der Erprobungsphase.

4.5 Bluetooth

Bluetooth wurde für die kabellose Kurzstreckenübertragung von Daten ähnlich Infrarotsystemen konzipiert. Allerdings hat es den Vorteil, dass es durch seine Technik auch durch Hindernisse hindurch Geräte miteinander verbinden kann. So können ohne Sichtkontakt zwischen den Geräten durch Funktechnik Daten übertragen werden. Notebook und Handy oder Digicam lassen sich so ohne Probleme verbinden. Auch innovative Ideen wie zum Beispiel intelligente Kleidung lässt sich damit spielend vernetzen.

4.6 VoiceXML

VoiceXML basiert auf dem XML-Standard. Mit ihm können durch spezielle "Voice"-Elemente Teile von Webseiten verarbeitet werden, um sie anschließend über eine "Voice Response Unit" in gesprochener Sprache wiederzugeben.

Durch die Beschränkungen heutiger Handys ist dies ein idealer Ersatz für die kleinen Tastaturen oder Displays. Es müssen nun nicht mehr umständliche Eingaben auf der Tastatur gemacht werden. Stattdessen spricht man einen Befehl ins Handy und bekommt durch VoiceXML die gewünschten Ergebnisse vorgelesen.

4.7 i-mode

i-mode ist ein von NTT DoCoMo entwickelter Übertragungsstandard ähnlich wie WAP und wurde 1999 in Japan eingeführt. Er überträgt Daten zwar auch mit 9600 Bit/s, ist aber inkompatibel zum GSM-Netz und benutzt paketorientierte Datenübermittlung. D.h., der Surfer kann ständig online sein, muss aber nur die tatsächlich übertragene Datenmenge bezahlen. Weiterhin programmiert man bei i-mode in cHTML, welches eine abgespeckte Variante von HTML 3.0 ist. Außerdem existieren schon Farbdisplays für Geräte, die diesen Standard unterstützen

und es können animierte Grafiken dargestellt werden. i-mode ist wegen diesen Möglichkeiten gegenüber WAP sehr erfolgreich und hat in Japan über 15 Millionen Nutzer. In anderen Ländern wird dieser Dienst zur Zeit noch nicht angeboten.

Kapitel 5

Zusammenfassung

Als vor einigen Jahren die ersten mobilen Telefongeräte auf den Markt kamen, ähnelten sie eher Feldtelefonen und konnten wegen der ungenügenden Netzabdeckung nur in bestimmten Gebieten genutzt werden. Das Telefonieren, wenn es denn mal gelang, war außerdem viel zu teuer für die Allgemeinheit. Die meisten Personen hatten aus diesem Grund kein Interesse am mobilen Telefonieren. Vielen langte es, das Telefon zu Hause zu benutzen.

Das gleiche spielt sich zur Zeit mit WAP ab. Es gibt oft Einwahlprobleme in das Internet, die WAP-Minutenpreise sind viel zu hoch und eigentlich kann auch von zu Hause aus gesurft werden. Doch werden sich die meisten Menschen in zehn Jahren fragen, wie sie auf den mobilen Internetanschluss verzichten konnten, genauso, wie mancher sich jetzt fragt, wie er ohne Handys auskommen konnte.

Wer regelmäßig mobil surft, kennt die schlechte Qualität vieler WAP-Services. Natürlich gibt es auch wohldurchdachte Konzepte, doch meist sieht es so aus, als ob aus dem Webangebot nur die Grafiken herausgenommen wurden. Die meisten Angebote glänzen außerdem nicht gerade durch effektive Benutzerführung. Auch der Inhalt enthält noch viel Nutzloses. Kein Wunder, weshalb viele Benutzer eines WAP-tauglichen Gerätes dieses nicht zum mobilen Surfen benutzen. Hier muss sich noch viel ändern.

Mit meiner Anleitung zu einem vernünftigen WAP-Service habe ich den Versuch unternommen, neben der Vermittlung von Grundlagenwissen über WAP das dynamische Programmieren von WML-Seiten zu demonstrieren. Anhand von

Beispielen ist es dem Programmierer möglich, schnell einen Einstieg in die WML-Programmierung zu bekommen. Dabei wurde viel Wert auf eine Benutzerfreundlichkeit auf dem WAP-Gerät gelegt. Weiterhin wurde für eine effektive Benutzerführung das Sessionmanagement von PHP benutzt und die Anbindung an eine MySQL-Datenbank erklärt.

Ich hoffe, ich konnte mit meiner Arbeit dazu beitragen, dass in Zukunft mehr attraktive mobile Angebote entwickelt werden. Insbesondere sollten die Programmierer darauf achten, gewisse Regeln in der Benutzerfreundlichkeit und im Bedienkomfort einzuhalten.

Literaturverzeichnis

- [1] WAP-Standards: www.wapforum.com
- [2] WML-Forum: groups.yahoo.com/group/wmlprogramming/
- [3] Watson, Karli: Beginning WAP, WML, and WMLScript, WROX Press, 2000, 1. Auflage
- [4] Arehart, Charles; Chidambaram, Nirmal; Guruprasad, Shashi u.a.: Professional WAP, WROX Press, 2000, 1. Auflage
- [5] Ziegler, Thomas: Einführung in WAP und WML, MITP-Verlag, 2000, 1. Auflage
- [6] Immler, Christian; Kreinacke, Markos; Spallek, Andre: WAP, DATA BECKER, 2000, 1. Auflage
- [7] Behme, Hennig: Aller Anfang, Apache für Linux mit PHP und Perl konfigurieren, IX, Heft 6, Juni 2000, Seite 48
- [8] Stoll, Rolf. D; Leierer, Gudrun Anna: PHP4 + MySQL, DATA Becker, 2000, 1. Auflage
- [9] PHP Dokumentation und Beispiele: www.php.net
- [10] WAP-F.A.Q: www.allnetdevices.com/faq/
- [11] MySQL Dokumentation: www.mysql.com/doc/
- [12] pic_2_wbmp: <http://www.gingco.de/wap/>
- [13] Terraflops Online Converter: <http://www.tera-flops.com/wbmp/>

Anhang A

Der Programmcode

A.1 index.php4

```
<?
  // Index page
  header("Content-type: text/vnd.wap.wml");
  echo("<?xml version=\"1.0\"?>\n");
  echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
    \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>
<wml>
<card id="c1" title="PasswortDB">
<p align="center">
  
</p>
<p>
  Passwort-<br/>datenbank
  <br/><a href="login.php4">Login</a>
  <br/><a href="newlogin.php4">Neues Login</a>
  <br/><a href="info.php4">Mehr Infos</a>
</p>
</card>
</wml>
```

A.2 config.php4

```
<?php
error_reporting(0); // 0 = none, 63 = full

//DB-Information
//insert your DB configuration here
$database = "1234";
$username = "1234";
$password = "1234";
$hostname = "db.123.de";

$maxlinks=10; // Sets amount of links per page
// nothing else needs to be configured below

// some usefull functions
// Output Header with no caching
function headerNoCache() {
    // set the correct MIME type
    header("Content-type: text/vnd.wap.wml");
    // expires in the past
    header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
    // Last modified, right now
    header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
    // Prevent caching, HTTP/1.1
    header("Cache-Control: max-age=0");
    // Prevent caching, HTTP/1.0
    header("Pragma: no-cache");
    // Set Document Type Definition (DTD)
    echo("<?xml version=\"1.0\"?>\n");
    echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
        \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
}

// Check, if a user is already in DB and return UUID
```

```
function checkAuth($user, $pwd, $db) {
    // Query Database
    $result = mysql_query("SELECT UUID FROM passuser
        WHERE user='$user' AND pwd='$pwd'", $db);
    // If no value, output error message
    $row = mysql_fetch_row($result) or die("
        <wml>
        <card id=\"c1\" title=\"Login\">
        <p>
        Die eingegebenen Login-Informationen
        sind inkorrekt. Bitte versuchen Sie es erneut.<br/>
        <a href=\"index.php4\">Zur&#252;ck</a>
        </p>
        <do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
        </card>
        </wml>");
    return $row[0];
}

// Convert Text to Unicode
function string2unicode ($text) {
    for ($i=0;$i<strlen($text);$i++) {
        $a=substr($text,$i,1); // Extract one single Character
        // change special chars above dec 126 to Unicode
        if (ord($a) > 126) {
            $text = substr_replace ($text, "&#".ord($a).";", $i, 1);
            $i=$i+5;
        }
        // Delete special chars below dec. 32 and above 255
        if ((ord($a) < 32) || (ord($a) > 255)) {
            $text = substr_replace($text, " ", $i, 1);
        }
        // Additionally change some other characters
        if ($a == "\\") {
            $text = substr_replace ($text, "&#".ord($a).";", $i, 1);
        }
    }
}
```

```

        $i=$i+4;
    };
    if ($a == "\"") {
        $text = substr_replace ($text, "&#.ord($a).\"", $i, 1);
        $i=$i+4;
    };
    if ($a == "<") {
        $text = substr_replace ($text, "&#.ord($a).\"", $i, 1);
        $i=$i+4;
    };
    if ($a == ">") {
        $text = substr_replace ($text, "&#.ord($a).\"", $i, 1);
        $i=$i+4;
    };
    if ($a == "&") {
        $text = substr_replace ($text, "&#.ord($a).\"", $i, 1);
        $i=$i+4;
    };
    if ($a == "\$") { // $ must be $$
        $text = substr_replace ($text, "\$\$", $i, 1);
        $i=$i+1;
    };
}
return $text;
}
?>

```

A.3 info.php4

```

<?
// Information Page
header("Content-type: text/vnd.wap.wml");
echo("<?xml version=\"1.0\"?>\n");
echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"

```

```
        \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
include(\"./config.php4\");
?>

<wml>
<card id=\"c1\" title=\"Information\">
<p>
    <? echo string2unicode(\"Hier können Sie
    komfortabel Ihre Passwörter oder
    sonstigen Daten ablegen und verwalten.
    Legen Sie wichtige Daten nie im Klartext ab!
    Der Übertragungsweg zur Datenbank ist vollkommen unsicher.
    Die Benutzung geschieht auf eigene Gefahr.
    Die Anwendung dient nur zu Lehrzwecken.
    Kontakt: bluesman@rz.tu-ilmenau.de
    \")?>
    <br/><a href=\"index.php4\">Zur&#252;ck</a>
</p>
<do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
</card>
</wml>
```

A.4 login.php4

```
<?
    // Login
    session_register (\"user\");
    session_register (\"pwd\");
    session_destroy(); // Destroy old sessions
    include(\"./config.php4\");
    headerNoCache(); // output header information
?>
<wml>
<card id=\"c1\" title=\"Login\" newcontext=\"true\">
```

```

<p>
  User:<br/>
  <input name="user" maxlength="15" value=""/><br/>
  PWD:<br/>
  <input name="pwd" maxlength="15" type="password" value=""/>
  <br/>
  <anchor>Login<go method="post" href="links.php4">
    <postfield name="user" value="$(user)"/>
    <postfield name="pwd" value="$(pwd)"/>
  </go>
</anchor>
<br/><a href="index.php4">Zur&#252;ck</a>
</p>
<do type="prev" label="Zur&#252;ck"><prev/></do>
</card>
</wml>

```

A.5 newlogin.php4

```

<?
  // Enter new login
  session_register ("user");
  session_register ("pwd");
  session_destroy(); // Destroy old sessions
  include("../config.php4");
  headerNoCache(); // output header information
?>

<wml>
<card id="c1" title="Neues Login">
<p>
  User:<br/>
  <input name="user" maxlength="15"/><br/>
  PWD:<br/>

```

```

<input name="pwd" maxlength="15" type="password" /><br/>
<anchor>Login<go method="post" href="newlogindo.php4">
    <postfield name="user" value="$(user)" />
    <postfield name="pwd" value="$(pwd)" />
    </go>
</anchor>
<br/><a href="index.php4">Zur&#252;ck</a>
</p>
<do type="prev" label="Zur&#252;ck"><prev/></do>
</card>
</wml>

```

A.6 newlogindo.php4

```

<?
// Check new Account data, if valid, insert in DB
error_reporting(0); // 0 = none, 63 = full
include("../config.php4");

// Debug Information and initialisation
if(!function_exists("mysql_result")) die("<center><b>Error -
function 'mysql_result' undefined please check for
MySQL in your server install</b></center>");

session_register ("user");
session_register ("pwd");
headerNoCache();

// Initialize connection to DB
$db = mysql_pconnect($hostname, $username, $password);
mysql_select_db($database, $db);
$result = mysql_query("SELECT UUID FROM passuser
    WHERE user='$user' ", $db);
echo "

```

```

        <wml>
        <card id=\"c1\" title=\"Logineintrag\">
        <p>;

$row = mysql_fetch_row($result);
if ($row[0] != "") {
    echo "User schon vorhanden.
        Bitte w&#228;hlen Sie einen neuen Usernamen.
        <br/><a href=\"newlogin.php4\">Zur&#252;ck</a>
    ";
} else {
    if (($user != "") AND ($pwd != "")) {
        $mysqltime = date("Y-m-d H:i:s");
        mysql_query("insert into passuser
            values(\"\", \"$user\", \"$pwd\", \"\".
                $mysqltime.\"\"), $db) or die("
        <wml>
        <card id=\"c1\" title=\"Login\">
        <p>
        Die eingegebenen Login-Informationen
        verursachten einen Fehler.
        Bitte w&#228;hlen Sie neue Zugangsdaten.<br/>
        <a href=\"newlogin.php4\">Zur&#252;ck</a>
        </p>
        <do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
        </card>
        </wml>");
    echo "Die Zugangsdaten wurden erfolgreich gespeichert.
        <br/><a href=\"links.php4\">Weiter</a>
    ";
} else { // User or Pwd empty
    echo "Username oder Passwort ist leer.
    <br/><a href=\"newlogin.php4\">Zur&#252;ck</a>";
};
};

```

```
echo "  
    </p>  
    </card>  
    </wml>";
```

A.7 links.php4

```
<?  
    // Checks Authentication, if valid, show links  
    error_reporting(0); // 0 = none, 63 = full  
    include("../config.php4");  
  
    // Debug Information and initialisation  
    if(!function_exists("mysql_result")) die("<center><b>Error -  
        function 'mysql_result' undefined please check  
        for MySQL in your server install</b></center>");  
  
    session_register ("user");  
    session_register ("pwd");  
    headerNoCache();  
  
    // Initialize connection to DB  
    $db = mysql_pconnect($hostname, $username, $password);  
    mysql_select_db($database, $db);  
    $uuid=checkAuth($user, $pwd, $db);  
  
    // Login sucessfull  
    if (!isset($lowlimit)) $lowlimit=0;  
    echo "  
        <wml>  
        <card id=\"c1\" title=\"Auswahl\">  
        <p>";  
        $mysqltime = date("Y-m-d H:i:s");
```

```

mysql_query("update passuser set lastaccess='$mysqltime'
WHERE uuid='$uuid' ", $db);
$result = mysql_query("SELECT link FROM passdata
WHERE uuid='$uuid' ORDER BY link LIMIT $lowlimit, ".
($maxlinks+1), $db);
$a=0;
$shownext=false;
while ($rowlink = mysql_fetch_row($result))
{
echo "<a href=\"text.php4?link=" . ($a+$lowlimit) . "\">"
.string2unicode($rowlink[0]) . "</a>";
echo "<br/>\n";
$a++;
$shownext=false;
if ($a == $maxlinks) {
// Check, if more links on another page are necessary
$rowlink = mysql_fetch_row($result);
if ($rowlink[0] != "") $shownext=true;
}
}
if ($a == 0)
echo "Noch keine Eintr&#228;ge vorhanden.<br/>";
if ($shownext)
echo "<a href=\"links.php4?
lowlimit=" . ($lowlimit+$maxlinks) . "\">Mehr Links
</a><br/>";
if ($lowlimit !=0)
echo "<a href=\"links.php4?
lowlimit=" . ($lowlimit-$maxlinks) . "\">Zur&#252;ck
</a><br/>";
echo "<a href=\"insert.php4\">Neuer Eintrag</a><br/>";
echo "<a href=\"index.php4\">Logout</a>
</p>
<do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
</card>

```

```
        </wml>" ;
?>
```

A.8 text.php4

```
<?
// Checks Authentication, if valid, show text of link
error_reporting(0); // 0 = none, 63 = full
include("../config.php4");

// Debug Information and initialisation

session_start();
headerNoCache();

// Initialize connection to DB
$db = mysql_pconnect($hostname, $username, $password);
mysql_select_db($database,$db);
$uuid=checkAuth($user, $pwd, $db);

// Login sucessfull
echo "
        <wml>
        <card id=\"c1\" title=\"Information\">
        <p>";
        $result = mysql_query("SELECT data FROM passdata
                WHERE uuid='$uuid' ORDER BY link",$db);
        $a=0;
        while ($rowdata = mysql_fetch_row($result)) {
                if ($a==$link) {
$text1=string2unicode($rowdata[0]);
break;
        };
        $a++;
```

```

    };
    echo $text1;
    echo "<br/><a href=\"links.php4\">Zur&#252;ck</a>";
    echo "<br/><a href=\"deldata.php4?
        link=$link\">Eintrag l&#246;schen</a>";
    echo "
</p>
<do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
</card>
</wml>";
?>

```

A.9 insert.php4

```

<?
// Authentication, if valid, allow to enter new link and data
error_reporting(0); // 0 = none, 63 = full
include("../config.php4");

// Debug Information and initialisation

session_start();
headerNoCache();

// Initialize connection to DB
$db = mysql_pconnect($hostname, $username, $password);
mysql_select_db($database,$db);
$uuid=checkAuth($user, $pwd, $db);

// Login sucessfull
echo "
    <wml>
    <card id=\"c1\" title=\"Dateneingabe\">
    <p>

```

```

Name des Eintrags:<br/>
<input name=\"link\" maxlength=\"15\" value=\"\"/><br/>
Daten:<br/>
<input name=\"data\" maxlength=\"500\"/><br/>
<anchor>&#220;bernehmen<go method=\"post\"
    href=\"insertcheck.php4?PHPSESSID=\";
    echo session_id();
    echo \"\">
    <postfield name=\"link\" value=\"\$(link)\"/>
    <postfield name=\"data\" value=\"\$(data)\"/>
</go>
</anchor>
<br/><a href=\"links.php4\">Abbruch</a>
</p>
<do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
</card>
</wml>

";
?>

```

A.10 insertcheck.php4

```

<?
// Checks entered link and data
error_reporting(0); // 0 = none, 63 = full
include("../config.php4");

// Debug Information and initialisation
session_start();
headerNoCache();

// Initialize connection to DB
$db = mysql_pconnect($hostname, $username, $password);
mysql_select_db($database, $db);

```

```
$uuid=checkAuth($user, $pwd, $db);

// Login sucessfull
echo "
    <wml>
    <card id=\"c1\" title=\"Dateneintrag\">
    <p>";
if (($link != "") AND ($data != "")) {
    mysql_query("insert into passdata
        values(\"\", \"$uuid\", \"$link\", \"$data\"),$db)
    or die("
    Die eingegebenen Daten
    verursachten einen internen Fehler.<br/>
    <a href=\"insert.php4\">Zur&#252;ck</a>
    </p>
    <do type=\"prev\" label=\"Zur&#252;ck\"><prev/></do>
    </card>
    </wml> ");
    echo "Die Daten wurden erfolgreich gespeichert.
    <br/><a href=\"links.php4\">Weiter</a>
    ";
} else { // User or Pwd empty
    echo "Name oder Datenfeld ist leer.
    <br/><a href=\"insert.php4\">Zur&#252;ck</a>";
};
echo "
    </p>
    </card>
    </wml>";
?>
```

A.11 deldata.php4

```
<?
// Checks Authentication, if valid, delete link and data
error_reporting(0); // 0 = none, 63 = full
include("../config.php4");

// Debug Information and initialisation

session_start();
headerNoCache();

// Initialize connection to DB
$db = mysql_pconnect($hostname, $username, $password);
mysql_select_db($database, $db);
$uuid=checkAuth($user, $pwd, $db);

// Login successful
echo "
    <wml>
    <card id=\"c1\" title=\"Dateneintrag\">
    <p>;
    $a=0;
    $result = mysql_query("SELECT id FROM passdata
        WHERE uuid=' $uuid' ORDER BY link", $db);
    while ($rowid = mysql_fetch_row($result)) {
        if ($a==$link) {
            mysql_query("DELETE FROM passdata
                WHERE id=' $rowid[0]'", $db) or die("
                Allgemeiner Datenbankfehler.<br/>
                <a href=\"links.php4\">Zur&#252;ck</a>
            </p></card></wml> ");
            break;
        };
        $a++;
    };
```

```
        };
        echo "Der Eintrag wurde erfolgreich gel&#246;scht.
        <br/><a href=\"links.php4\">Weiter</a>
    ";
    echo "
        </p>
        </card>
        </wml>";
?>
```

A.12 maketable.sql

```
CREATE TABLE passuser (
    UUID int(10) NOT NULL AUTO_INCREMENT,
    user varchar(15) NOT NULL,
    pwd varchar(15) NOT NULL,
    lastaccess datetime,
    PRIMARY KEY (UUID)
);

CREATE TABLE passdata (
    id int(10) NOT NULL AUTO_INCREMENT,
    UUID int(10) NOT NULL,
    link varchar(20) NOT NULL,
    data text,
    PRIMARY KEY (id)
);
```